

UNIT05:SPRITE – TEXTURE

【学習要項】

- ☐DirectXTK
- ☐Texture space
- ☐Texel space
- ☐Sampler state

【演習手順】

1. 画像ファイルのロードのために DirectXTK を利用する
 - ①DirectXTK を GitHub からダウンロード (<https://github.com/Microsoft/DirectXTK>)
 - ※「Code」から「Download ZIP」を選択
 - ②ダウンロードした ZIP を x3dgp.00 プロジェクトフォルダに展開
 - ③VisualStudio のメニュー「ファイル」→「追加」→「既存のプロジェクト」を選択
 - ④展開した DirectXTK のフォルダから DirectXTK_Desktop_2019.vcxproj を選択
 - ⑤メニュー「プロジェクト」→「プロジェクト依存関係」を選択し、DirectXTK_Desktop_2019 をチェック
 - ⑥ソリューションエクスプローラのソリューション'3dgp'を右クリックし、プロパティを選択
 - ⑦3dgp プロパティページの「C/C++」→「全般」→「追加のインクルード…」に以下のパスを追加
.<¥DirectXTK-master¥Inc
 - ⑧3dgp プロパティページの「リンカー」→「全般」→「追加のライブラリ…」に以下のパスを追加
.<¥DirectXTK-master¥Bin¥x64¥Debug
 - ⑨3dgp プロパティページの「リンカー」→「入力」→「追加の依存…」に DirectXTK.lib を追加

2. sprite クラスに以下のメンバ変数を追加する

```
ID3D11ShaderResourceView* shader_resource_view;  
D3D11_TEXTURE2D_DESC texture2d_desc;
```

3. sprite クラスの vertex 構造体にテクスチャ座標変数を追加する

```
1: struct vertex  
2: {  
3:     DirectX::XMFLLOAT3 position;  
4:     DirectX::XMFLLOAT4 color;  
*5:     DirectX::XMFLLOAT2 texcoord;  
6: };
```

4. sprite クラスのコンストラクタのインターフェイスを変更する

```
sprite(ID3D11Device* device, const wchar_t* filename);
```

5. sprite クラスのコンストラクタの実装を変更する

- ① vertex 構造体の初期値を追加変更する

```
1: vertex vertices[]  
2: {  
*3:     { { -1.0, +1.0, 0 }, { 1, 1, 1, 1 }, { 0, 0 } },  
*4:     { { +1.0, +1.0, 0 }, { 1, 1, 1, 1 }, { 1, 0 } },  
*5:     { { -1.0, -1.0, 0 }, { 1, 1, 1, 1 }, { 0, 1 } },  
*6:     { { +1.0, -1.0, 0 }, { 1, 1, 1, 1 }, { 1, 1 } },  
7: };
```

- ②入力レイアウト要素の追加

```
1: D3D11_INPUT_ELEMENT_DESC input_element_desc[]  
2: {  
3:     { "POSITION", 0, DXGI_FORMAT_R32G32B32_FLOAT, 0,  
4:     D3D11_APPEND_ALIGNED_ELEMENT, D3D11_INPUT_PER_VERTEX_DATA, 0 },  
5:     { "COLOR", 0, DXGI_FORMAT_R32G32B32A32_FLOAT, 0,  
6:     D3D11_APPEND_ALIGNED_ELEMENT, D3D11_INPUT_PER_VERTEX_DATA, 0 },  
*7:     { "TEXCOORD", 0, DXGI_FORMAT_R32G32_FLOAT, 0,  
*8:     D3D11_APPEND_ALIGNED_ELEMENT, D3D11_INPUT_PER_VERTEX_DATA, 0 },  
9: };
```

- ③画像ファイルのロードとシェーダーリソースビューオブジェクト(ID3D11ShaderResourceView)の生成

```
※依存するインクルードファイル  
#include <WICTextureLoader.h>
```

```
1: ID3D11Resource* resource{};
2: hr = DirectX::CreateWICTextureFromFile(device, filename, &resource, &shader_resource_view);
3: _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
4: resource->Release();
```

④テクスチャ情報(D3D11_TEXTURE2D_DESC)の取得

```
1: ID3D11Texture2D* texture2d{};
2: hr = resource->QueryInterface<ID3D11Texture2D>(&texture2d);
3: _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
4: texture2d->GetDesc(&texture2d_desc);
5: texture2d->Release();
```

5. sprite クラスの render メンバ関数の実装を変更する

①テクスチャ座標を頂点バッファにセットする

※今回はテクスチャ全体を表示する

```
1: HRESULT hr{ S_OK };
2: D3D11_MAPPED_SUBRESOURCE mapped_subresource{};
3: hr = immediate_context->Map(vertex_buffer, 0, D3D11_MAP_WRITE_DISCARD, 0, &mapped_subresource);
4: _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
5:
6: vertex* vertices{ reinterpret_cast<vertex*>(mapped_subresource.pData) };
7: if (vertices != nullptr)
8: {
9:     vertices[0].position = { x0, y0 , 0 };
10:    vertices[1].position = { x1, y1 , 0 };
11:    vertices[2].position = { x2, y2 , 0 };
12:    vertices[3].position = { x3, y3 , 0 };
13:    //vertices[0].color = vertices[1].color = vertices[2].color = vertices[3].color = { 1, 1, 1, 1 };
14:    vertices[0].color = vertices[1].color = vertices[2].color = vertices[3].color = { r, g, b, a };
15:
16:    vertices[0].texcoord = { 0, 0 };
17:    vertices[1].texcoord = { 1, 0 };
18:    vertices[2].texcoord = { 0, 1 };
19:    vertices[3].texcoord = { 1, 1 };
20: }
21:
22: immediate_context->Unmap(vertex_buffer, 0);
```

②シェーダー リソースのバインド

```
immediate_context->PSSetShaderResources(0, 1, &shader_resource_view);
```

6. sprite.hlsl の VS_OUT 構造体を変更する

```
1: struct VS_OUT
2: {
3:     float4 pos : SV_POSITION;
4:     float4 color : COLOR;
5:     float2 texcoord : TEXCOORD;
6: };
```

7. sprite_vs.hlsl を変更する

```
*1: VS_OUT main(float4 position : POSITION, float4 color : COLOR, float2 texcoord : TEXCOORD)
2: {
3:     VS_OUT vout;
4:     vout.position = position;
5:     vout.color = color;
6:
7:     vout.texcoord = texcoord;
8:
9:     return vout;
```

```
10: }
```

8. sprite_ps.hlsl を変更する

```
1: #include "sprite.hlsl"
2: Texture2D color_map : register(t0);
3: SamplerState point_sampler_state : register(s0);
4: SamplerState linear_sampler_state : register(s1);
5: SamplerState anisotropic_sampler_state : register(s2);
6: float4 main(VS_OUT pin) : SV_TARGET
7: {
8:     return color_map.Sample(point_sampler_state, pin.texcoord);
9: }
```

9. framework クラスの initialize メンバ関数での sprite オブジェクトの生成方法を変更する

```
sprites[0] = new sprite(device, L"¥¥resources¥¥cyberpunk.jpg");
```

10. サンプラス状態オブジェクト(ID3D11SamplerState)

① framework クラスのメンバ変数として以下のコードを追加する

```
ID3D11SamplerState* sampler_states[3];
```

② framework クラスの initialize メンバ関数でサンプラス状態オブジェクトを生成する

```
1: D3D11_SAMPLER_DESC sampler_desc;
2: sampler_desc.Filter = D3D11_FILTER_MIN_MAG_MIP_POINT;
3: sampler_desc.AddressU = D3D11_TEXTURE_ADDRESS_WRAP;
4: sampler_desc.AddressV = D3D11_TEXTURE_ADDRESS_WRAP;
5: sampler_desc.AddressW = D3D11_TEXTURE_ADDRESS_WRAP;
6: sampler_desc.MipLODBias = 0;
7: sampler_desc.MaxAnisotropy = 16;
8: sampler_desc.ComparisonFunc = D3D11_COMPARISON_ALWAYS;
9: sampler_desc.BorderColor[0] = 0;
10: sampler_desc.BorderColor[1] = 0;
11: sampler_desc.BorderColor[2] = 0;
12: sampler_desc.BorderColor[3] = 0;
13: sampler_desc.MinLOD = 0;
14: sampler_desc.MaxLOD = D3D11_FLOAT32_MAX;
15: hr = device->CreateSamplerState(&sampler_desc, &sampler_states[0]);
16: _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
17:
18: sampler_desc.Filter = D3D11_FILTER_MIN_MAG_MIP_LINEAR;
19: hr = device->CreateSamplerState(&sampler_desc, &sampler_states[1]);
20: _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
21:
22: sampler_desc.Filter = D3D11_FILTER_ANISOTROPIC;
23: hr = device->CreateSamplerState(&sampler_desc, &sampler_states[2]);
24: _ASSERT_EXPR(SUCCEEDED(hr), hr_trace(hr));
```

③ framework クラスの render メンバ関数でサンプラス状態オブジェクトをバインドする

```
1: immediate_context->PSSetSamplers(0, 1, &sampler_states[0]);
2: immediate_context->PSSetSamplers(1, 1, &sampler_states[1]);
3: immediate_context->PSSetSamplers(2, 1, &sampler_states[2]);
```

11. 実行し、キャラクタの画像が表示することを確認する

※出力ウィンドウに COM オブジェクト未開放の警告が出るので、適切な場所で解放する

12. 頂点カラーの値が反映するようにピクセルシェーダを変更する

13. 使用するサンプラス状態を変更しての描画結果の違いを確認する

【評価項目】

- ☐ロードされた画像全体が画面上の任意の位置・サイズ・回転量で表示
- ☐頂点カラーの反映
- ☐サンプラス状態の効果